On Motion Coordination of Robots with Volume

T. Minami
H. Kakugawa
Ichiro Suzuki
Masafumi Yamashita

TR-94-02-03
February 14, 1994

Department of Electrical Engineering and Computer Science
University of Wisconsin – Milwaukee

On Motion Coordination of Robots with Volume*

T. Minami†, H. Kakugawa†, I. Suzuki‡ and M. Yamashita†

†Department of Electrical Engineering
Faculty of Engineering
Hiroshima University

‡Department of Electrical Engineering and Computer Science
University of Wisconsin–Milwaukee

February 14, 1994

1 Introduction

In the study of cooperative behavior of distributed autonomous robots, certain problems including convergence to a single point and formation of the given geometric figures have been discussed under a simplifying assumption that a robot is represented as a point that has no volume [1]. In this paper, we represent each robot as a disc with some volume, rather than a point, and develop and evaluate two algorithms for moving many robots through a narrow gate. Though each robot is a disc, we assume that a robot does not block the view of other robots.

The gate can be thought of as a narrow slit on the otherwise solid y-axis of the x-y plane, where the robots (discs) are trying to move from the left of the y-axis to the right.

Aside from the assumption that a robot is a disc, the model of the robot system we use is essentially the same as that given in [1]. Namely, we assume that: (1) A robot is a mobile processor that has a sensor for detecting the distances and directions of other robots. (2) The next position of a robot is determined by the given algorithm

*This work was supported in part by the National Science Foundation under grants CCR-9004346 and IRI-9307506, and by the Office of Naval Research under grant N00014-94-1-0284. This report is a revised version of the paper that appeared in *Proceedings of the 1993 Joint Symposium on Electronics and Information*, Institute of Electronics, Information and Communication Engineers (IEICE), Hiroshima, Japan, October 1993, page 337. A copy of the paper is attached.

using the relative positions of other robots. (3) All robots execute the same algorithm. (4) The robots do not have a common x-y coordinate system. (5) The robots do not have a sense of direction. (6) The robots are indistinguishable by their appearances.

2 The First Algorithm

2.1 Algorithm Right-Left-1

The basic structure of the algorithms we consider is the following: At intervals of unit time, each robot R observes the positions of other robots, computes the next position using the given algorithm, and moves to the position. One execution of this process is called a step.

To describe the movement of R, we partition the plane into six equal-sized wedges about the position of R (allotting the wedge boundaries appropriately), and call them R (right), RR (right-rear), LR (left-rear), L (left), LF (left-front), and RF (right-front) in clockwise order, starting from the wedge on the right of R. A robot is said to be in wedge R, for example, if its center is in R. The distance between two robots are measured between their centers.

Consider the following simple algorithm Right-Left-1 that each robot R uses to determine its motion in each step. It is assumed that robot R is always facing toward the center O of the gate. We use two parameters, δ and s, where $\delta > 1.0$ is a constant that (indirectly) determines how close two robots can approach each other, and $s > 0$ is the maximum distance that a robot can move in unit time. The phrase "R moves left" given below, for example, means that R moves left, over distance s if no other robots exists on the trajectory of R, but otherwise R stops immediately before the first collision with other robots. There are five cases, which are given in the decreasing order of precedence.

Case 1: If a robot exists in R $\cup$ RF within distance δ and no robot exists in L $\cup$ LF within distance δ, then R moves left (along the line perpendicular to the line that passes through its current position and O).

Case 2: This case is the same as **Case 1**, except that left and right are interchanged.

Case 3: If no robot exists in RF $\cup$ LF within distance δ, then R moves forward.

Case 4: If no robot exists in RR $\cup$ LR within distance δ, then R moves backward.

Case 5: If none of the above cases applies, then R does not move.

2.2 Outline of Simulation

We used simulation to evaluate the performance of algorithm Right-Left-1 given above, under the following conditions.

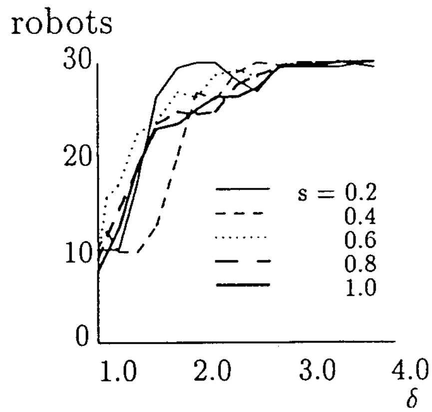

Figure 1: Right-Left-1: δ and the number of robots that have successfully passed through the gate.

1. Each robot is a disc with diameter 1.0. The total number of robots is $N = 30$. Initially, the robots are randomly and uniformly distributed within a semicircle of radius 30.0 centered at O.

2. The width of the gate is slightly less than 2.0.

3. The maximum distance that a robot can move in unit time is denoted s. We try various values of s in the range from 0.2 to 1.0.

4. We try various values of δ in the range from 1.1 to 3.8.

5. If no robot passes through the gate for $100/s$ steps consecutively, then we terminate the simulation assuming that the robots has reached a deadlock-like situation.

6. We execute 100 simulation runs for each combination of values of s and δ.

2.3 Simulation Results

The simulation results are shown in Figures 1 and 2. Figure 1 describes the relation between the value of δ and the total number of robots that have successfully passed through the gate. Figure 2 shows the total number of steps needed for all 30 robots to pass through the gate.

We observe that generally, as the value of δ gets larger, the total number of robots that pass through the gate increases, but at the same time, the total number of steps needed for all 30 robots to pass through the gate increases. One possible explanation

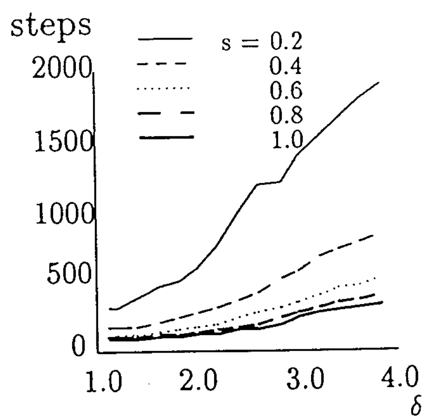

Figure 2: Right-Left-1: δ and the number of steps needed for all 30 robots to pass through the gate.

is that a larger value of δ will make the distribution of the robots more sparse, and thus the robots are less likely to reach a deadlock situation. So the chances that all 30 robots pass through the gate increases, but the total number of steps needed may increase since the inter-robot distances are now larger. So there is a tradeoff between the "success rate" and the total time needed for completion of the task.

Another observation is that algorithm Right-Left-1 is based on the general strategy of reducing the density of robot distribution, but once the robots get highly congested, the algorithm does not seem to provide any effective means to reduce the congestion quickly.

3 The Second Algorithm

3.1 Algorithm Right-Left-2

Based on the observation given at the end of the previous section, we modify Right-Left-1 by including an additional instruction that is expected to be effective in resolving the congestions of the robots. The resulting algorithm is called Right-Left-2, and is given below, again as a set of instructions to robot R:

Case 0: If R moved over distance less than s in the previous step (to avoid collision with other robots), and a robot exists in $\mathsf{R} \cup \mathsf{RF} \cup \mathsf{L} \cup \mathsf{LF}$ within distance δ, then R moves in the direction away from the robot that is closest to R.

Cases 1–5: These cases are identical to those in Right-Left-1.

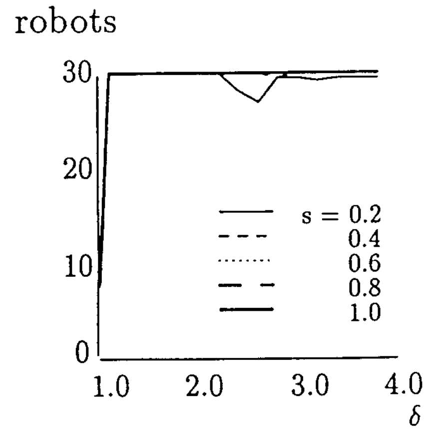

Figure 3: Right-Left-2: δ and the number of robots that have successfully passed through the gate.

3.2 Simulation Results

We conducted a series of simulation runs to evaluate Right-Left-2, under the conditions similar to those described in **2.3** for Right-Left-1. The results are shown in Figures 3 and 4. As we see in Figure 3 except for some very small values of δ, all 30 robots can pass through the gate with high probability. (See Figure 2 for comparison with Right-Left-1.) Since the robots behave in the same manner in both algorithms unless the robots are congested and collision occurs, the improvement in performance of Right-Left-2 over Right-Left-1 must be due to the ability of Right-Left-2 to resolve congestion and allow the robot to move further. The total number of steps needed to move all 30 robots through the gate seems to be nearly the same for both algorithms (Figures 2 and 4).

4 Conclusion

We found that the modified algorithm allows all 30 robots to pass through the gate with high probability. The basic strategy used in our algorithms is rather simple—each robot individually tries to move toward the gate avoiding other robots by swerving either to the right or left. We are planning to examine other strategies, such as forming a line distributively, in order to find a better algorithm that can move the robots through the gate faster with very high probability.

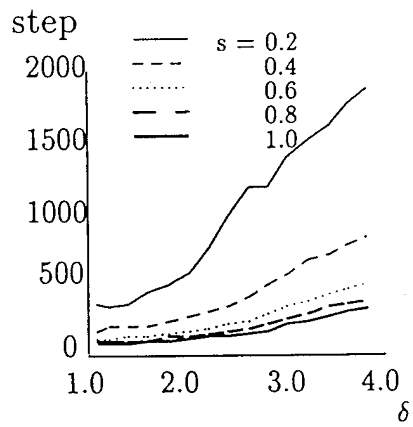

Figure 4: Right-Left-2: δ and the number of steps needed for all 30 robots to pass through the gate.

References

[1] I. Suzuki and M. Yamashita, "Cooperative control algorithms for anonymous mobile robots," in *Proceedings of the 6th Karuizawa Workshop on Circuits and Systems*, April 1993, pp. 525–530.

121901

大きさを持つロボットの協調について

南 健† 角川 裕次† 鈴木 一郎‡ 山下 雅史†
(†広島大学工学部) (‡U.wisconsin-Milwaukee)

1 はじめに

自律分散ロボットの協調動作に関する研究においては、ロボットを質点と仮定して幾何図形の形成や一点集中などの問題が提示されている [1]。この中でロボットの持つ能力として以下を仮定する: (1) 各ロボットは他のロボットの位置 (距離と方向) を確認できる、(2) 各ロボットは絶対座標を持たない、(3) 各ロボットはある特定の方向を決定できるコンパスを持たない、(4) 各ロボットは全て同じに見える、(5) 各ロボットは全て同じアルゴリズムを実行する。

本稿では、上の条件に加えて、各ロボットを大きさを持った円であると仮定し、狭い関門を通り抜ける問題に対する、アルゴリズムを提案し評価する。但し、ロボットは他のロボットの視界を妨げないものとする。

2 関門を通過するアルゴリズム

2.1 アルゴリズム

アルゴリズムの基本的構造は次の通りである: ロボットはある単位時間ごとに周辺のロボットの位置を確認し、アルゴリズムに従って移動目標を計算し移動する。これを 1 ステップと呼ぶ。

まず、次の単純なアルゴリズム Right-Left1 を考える。

Case 1: 他ロボットが右前と右横どちらかに存在して左前と左横両方に存在しなかったらゲートに向かって左側へ垂直に移動する。
Case 2: 左右入れ換えて Case 1 を行う。
Case 3: 他ロボットが前方に存在しなければ前方へ移動する。
Case 4: 他ロボットが後方に存在しなければ後方へ移動する。
Case 5: 上記いずれにも当てはまらない場合停止する。

ただし各 Case は上のものほど優先順位が高いとする。また '存在する' とは、あるロボットと他のロボットとの距離が δ 以下になった状態を指す。

2.2 実験

上のアルゴリズムに対し、以下の条件によるシミュレーション実験を行なった。

1. ロボットを直径 1.0 の円とする。ロボットの総数を $N = 30$ とし、初期状態としてゲートを中心とする半径 50.0 の半円内に一様かつランダムに分布させる。
2. ゲートの幅は 2.0 よりわずかに狭いとする。
3. 単位時間の最大移動距離 s を 0.2 から 1.0 まで、δ を 1.1 から 3.8 まで変化させる。
4. $100/s$ ステップ連続してロボットがゲートを通らない場合、ロボットがつまったとして終了する。
5. s と δ の組合せについてそれぞれ 100 回試行する。

2.3 結果

結果を図 1、2 に示す。図 1 は δ とゲートを通ったロボット数との関係を、図 2 は δ と 30 体目のロボットがゲートを通ったときのステップ数との関係を表している。

基本的に δ の値が大きいほどより多くのロボットがゲートを通っているが、30 体目がゲートを通るステップ数は増大している。これは δ が大きくなるとロボット間の距離が開くため、ロボットの密度が小さくなるからであると考えられる。

これから、ゲートを通るロボットの数を増やすためには δ を大きくする必要があるが、より時間がかかってしまう事が分かる。またこのアルゴリズムはロボットの密度を低くするという方策に従ったアルゴリズムであり、ロボットがつかえると、なかなかつかえた状態が解消されないと思われる。

3 アルゴリズムの改良

3.1 改良後のアルゴリズム

前述のアルゴリズム Right-Left1 は、ロボットがつかえた時の解消に問題がある。この点に対し改良を加えたアルゴリズム Right-Left2 を提案する。

Case 0: 直前のステップで他のロボットと衝突した場合 左前か左横に (あるいは右前か右横に) 他のロボットが存在すれば 一番近いロボットと半対側に移動する。
Case 1-5 は、Right-Left1 と同じ。

3.2 実験と結果

2.2 と同様の実験を行う。結果を図 3、4 に示す。図 3 より、δ の値がある程度以上になればかなり高い確率で全てのロボットがゲートをくぐることができることが分かる。これはアルゴリズムの改良によって、ロボットがつかえても再び動ける状態に戻ることが可能になったためと思われる。

4 おわりに

改良したアルゴリズムでは、かなり高い確率で全てのロボットがゲートをくぐりぬけるようになった。これからの課題としては、より確実により速くロボットがゲートを通るようなアルゴリズムの確立である。またこの左右に避けてゆくアルゴリズムは最も単純なアルゴリズムであるので、行列を作る等、他のアプローチについても検討してゆく予定である。

参考文献

[1] I. Suzuki and M. Yamashita, "Cooperative Control Algorithms for Anonymous Mobile Robots," 数理解析研究所講究録 833: 計算機構とアルゴリズム, 京都大学数理解析研究所. February 16, 1993, 142-152.

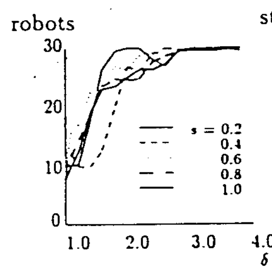

図 1 δ-number

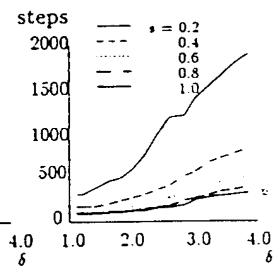

図 2 δ-steps

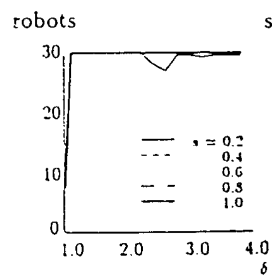

図 3 δ-number

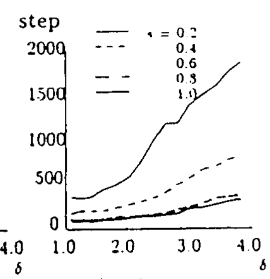

図 4 δ-step